Implementação de um software para geração semi-automática de grades de horário

Lucas dos Santos Morisso[1], Ricardo Fröhlich da Silva[2]

[1,2] Curso de Ciência da Computação – Área de Ciências Tecnológicas - Centro Universitário Franciscano – Rua dos Andradas, 1614 – Santa Maria (RS) – Brasil.

ls.morisso@gmail.com, ricardosma@gmail.com

Resumo

Este trabalho apresenta uma solução para o problema da grade de horários de uma escola de idiomas. O trabalho buscou, através da coleta de dados de entrada e com o auxílio de um software baseado em força bruta, realizar a alocação dos professores para a formulação dessa grade chegando a um resultado satisfatório. Estes foram baseados nos dados obtidos e informados na grade de horários criada na escola de idiomas, que por sua vez possui regras e restrições diferentes de uma instituição de ensino tradicional. Esse software pretende facilitar o trabalho dos responsáveis por esta tarefa, já que hoje soluções encontradas em outros softwares não são eficientes ou não resolvem por completo os requisitos exigidos pela empresa envolvida.

Abstract

This paper presents a solution to the problem of scheduling grid of a language school. The study sought, through data collection and input with the help of a software based on brute force, perform the allocation of teachers for the formulation of this grid reaching a satisfactory outcome. These were based on the data obtained and informed in the grid of timetables established in the school of languages, which in turn has different rules and constraints of a traditional educational institution. This software is intended to facilitate the work of those responsible for this exhausting task, since today in other software solutions are inefficient or don't solve completely the requirements demanded by the company involved.

1 Introdução

Ao início de cada ano letivo as instituições de ensino enfrentam um problema normalmente difícil de ser resolvido de forma manual. Esse problema está relacionado à necessidade da elaboração da grade de horários dos professores do quadro funcional, onde se faz necessário alocar os mesmos nas turmas existentes baseando-se nas normas e restrições que compõem o exercício desse profissional.

A resolutividade está diretamente relacionada com o indivíduo capacitado que, manualmente deverá distribuir corretamente os encargos didáticos aos professores, num trabalho que exige muita concentração e que demanda grande período de tempo. Infelizmente, na maior parte dos casos o trabalho realizado não oferece formulações de encargos corretas, fazendo com que professores e alunos sejam prejudicados, por

exemplo, com apenas uma hora aula de determinada disciplina ou com cinco horas aula seguidas.

Mediante tais questões, não se conhece algoritmos polinomiais para resolvê-lo na maioria das situações em que se apresenta, pois o problema de geração de grades de horário, também conhecido por *timetable*, é um problema de otimização que possui várias aplicações práticas, por exemplo, escalonamento de enfermeiros, horários de aula em instituições de ensino [Caldeira e Rosa, 1997] e planejamento de transporte público, devido ao fato de cada instituição ter seu modo de tratar os horários, não existindo uma ferramenta universal para tratamento do problema [Hamawaki, 2005].

Considerando que na escola em questão não há um número elevado de professores (atualmente treze professores) e moderado em restrições - considerando que há somente uma disciplina a ser ministrada (inglês), foi desenvolvida uma ferramenta com a utilização do método força bruta através de informações armazenadas em um banco de dados (MySql). O referido método trabalha por repetição procurando por todas as soluções possíveis e verificando se cada uma satisfaz as restrições e limitações da escola e professores.

A implementação desse algoritmo é muito dinâmica e sempre se achará uma solução (resultado), quando existente. Entretanto o seu custo e tempo computacional é proporcional ao número de possíveis soluções, que, em problemas reais, tende a crescer exponencialmente. Sendo assim, a força bruta é tipicamente usada em problemas cujo tamanho é limitado [Lourenço, Paixão e Portugal, 2001], enquadrando-se nas exigências da referida instituição.

Portanto, objetivou-se através das informações passadas por um usuário ao sistema, o resultado de uma grade de horários adequada aos professores e a instituição em questão gerada em tempo hábil e possível de ser utilizada.

2 Referencial Teórico

Em virtude do aumento de instituições com grande número de professores, bem como alunos em diferentes turmas (classes), um problema relevante se consolida: criar uma grade de horário que consiga alocar os docentes de modo que não ocorram conflitos de disciplinas, nem entre diferentes locais de trabalho, ao mesmo tempo em que deve respeitar as restrições impostas por cada instituição.

De acordo com [Fucilini *et al*. 2008] a maior complexidade nesta construção encontra-se marcada pela busca de uma solução exata para tais restrições. Para [Camargo *et al*. 2011] o principal obstáculo encontrado por quem organiza o quadro de horários é a complexidade em tentar não infringir as restrições que existem, seja pelas disponibilidades dos professores, ou pela política da organização. Quanto mais o quadro de horários se aproxima do fim do preenchimento, mais as restrições ficam evidentes e dificultam a finalização do processo com sucesso.

2.1 O *timetable* na complexidade do NP-Completo

No meio acadêmico este problema é conhecido como sendo da classe dos NP – Completo (complexidade computacional), o que significa que o *timetabling* não tem uma solução polinomial, porque mesmo tentando subdividí-lo em problemas menores, continua sendo da classe dos NP. A maioria dos teóricos acredita que os problemas NP completos são intratáveis, sem resultados conclusivos, pois, no momento em que um *timetable* se efetiva para dada instituição só será efetiva para outra diferente caso esta contemple todas as restrições da anterior [Cormen *et al*. 2002]. Existe uma dificuldade

em resolver os problemas combinatórios devido à diversidade e complexidade das restrições que devem ser satisfeitas em cada caso.

Devido à existência de limitações nos recursos oferecidos pelos computadores, identificar um problema e descobrir ao menos uma solução para este não é o suficiente, pois também é preciso avaliar se essa solução encontrada pode ser resolvida efetivamente em tempo e espaço de memória finitos [Aguiar, 1998]. Com este propósito as principais classes de complexidade para problemas são analisadas a seguir:

- **Classe P:** consiste nos problemas que podem ser resolvidos por um algoritmo determinístico em um tempo polinomial. Ela contém todos os problemas considerados computacionalmente tratáveis. Porém, um problema tratável não é necessariamente considerado eficiente, pois seu tempo de execução pode ser muito extenso.
- **Classe NP:** define-se por um conjunto de todos os problemas que podem ser resolvidos por um algoritmo não-determinístico em tempo polinomial. Onde, um algoritmo não-determinístico é considerado de tempo polinomial quando se observa que uma dada entrada satisfaz o problema em tempo polinomial.
- **Classe NP-Completo:** são problemas NP que possuem a característica de que se um deles puder ser resolvido em tempo polinomial então todo problema NP-Completo terá uma solução em tempo polinomial. As principais classes de complexidade foram descritas, para o auxílio, no desenvolvimento de algoritmos que busquem não apenas resultados satisfatórios como também dentro de um tempo válido [Aguiar, 1998].

Um exemplo clássico de NP-Completo é o caso do "Caixeiro Viajante" onde se tem por objetivo descobrir a menor rota para uma viagem, passando por um número 'c' de cidades e retornando a cidade de partida. Exemplificando, trata-se c = 4, onde o viajante terá de passar por quatro (4) cidades.

Para identificar o número de rotas **R(c)** é necessário fazer um raciocínio combinatório simples, onde tem-se por base que a primeira e última posição são fixas, pois é a cidade de onde o viajante sai e deve voltar. Assim, é possível verificar que o total de escolhas que podem ser feitas nesse caso é de (c-1) x (c-2) x (c-3) = 6. De modo que, usando a notação de fatorial **R(c) = (c – 1)!** é possível obter todas as rotas possíveis e assim calcular a distância entre as cidades e ver a rota que apresenta um comprimento total menor. Aparentemente um caso fácil de ser resolvido por um computador capaz de fazer 1bilhão de adições por segundo por exemplo.

Mas se ao invés de 4 fossem 20 cidades, o computador precisaria apenas de 19 adições para descobrir o comprimento de uma rota e seria capaz de calcular 109 / 19 = 53 milhões de rotas por segundo. Contudo, a imensidão do número **19!** de rotas para analisar tornam esta operação inviável, pois 19! é aproximadamente 1.2×10^{17} dividindo por 53 milhões de rotas por segundo, o resultado teria um tempo de execução de 2.3×10^{9} segundos aproximadamente cerca de 73 anos para o computador executar sua tarefa. Seguindo esse raciocínio surge à busca pela otimização, marcada como um mecanismo de análise de decisões complexas, envolvendo seleção de variáveis, com o simples objetivo de quantificar desempenho e medir a qualidade das decisões 16 [Soares, 1997, p.4]. Como citado acima, procura-se uma forma eficaz, simples e rápida, apesar de todas as informações envolvidas, para aperfeiçoar o tempo gasto pelos responsáveis pela criação destas grades de horários avaliando sua qualidade e principalmente sua eficiência ao obter os resultados.

2.2 Grade de horário (*timetable)*

Em virtude do aumento de instituições com grande número de professores, bem como alunos em diferentes turmas (classes), um problema relevante se consolida: criar uma grade de horário que consiga alocar os docentes de modo que não ocorram conflitos de disciplinas, nem entre diferentes locais de trabalho, ao mesmo tempo em que deve respeitar as restrições impostas por cada instituição.

No desenvolvimento da grade de horário na referida escola de idiomas, as questões problema levantadas primariamente foram aos poucos sendo trabalhadas e aprimoradas. Especificamente, a escola ofereceu algumas restrições importantes que foram consideradas. Conforme relatado anteriormente, o quadro funcional atual está composto por treze (13) docentes sendo esses avaliados (nivelados) a fim de considerá-los aptos para ministrar aulas em determinado nível. As turmas, por níveis, são abertas conforme a procura podendo ser iniciadas a qualquer instante, existindo um limite de turma por hora, limitando-se ao número de salas da instituição.

2.2.1 Método Força Bruta como algoritmo base para a construção de *timetabling*

De acordo com Quadros (2011), Força Bruta (ou busca exaustiva) é um algoritmo trivial, mas de uso muito geral que consiste em enumerar todos os possíveis candidatos de uma solução e verificar se cada um satisfaz o problema. Esse algoritmo possui uma implementação muito simples, e sempre encontrará uma solução se ela existir. Entretanto, seu custo computacional é proporcional ao número de candidatos à solução, que, em problemas reais, tende a crescer exponencialmente.

Apesar de ser raramente uma fonte de algoritmos eficientes ou brilhantes, a técnica força bruta é uma importante estratégia de projeto de algoritmos [Koerich, 2004] o que colaborou para a escolha desse método.

2.2.2 Algoritmos genéticos e sua aplicabilidade em resposta ao *timetabling*

Através do estudo sobre algoritmos de otimização, verificando-se, entre tantas, a característica de atingir mais vezes a solução global por número de execuções, os algoritmos genéticos encontram-se como um dos métodos mais eficazes [Soares, 1997/ Terra; Radaelli, 2007]. Algoritmos genéticos tratam-se de uma aplicação computacional baseada na teoria da evolução natural e na genética. A cada nova população esse algoritmo deve evoluir, tendendo chegar a um resultado ideal, porém ele é totalmente dependente do ambiente (regras de evolução, cruzamento, mutações, seleções). O desempenho do algoritmo genético é extremamente sensível ao tamanho dos dados oferecidos, caso este número seja pequeno demais, não haverá espaço para se ter variedade genética tornando assim o algoritmo incapaz de achar boas soluções [Linden, 2008].

Enfatizando tal colocação explica-se a opção pelo método força bruta sem a utilização de algoritmos genéticos, uma vez que para o desenvolvimento da grade de horários em questão o número de dados pode ser considerado pequeno inviabilizando a amplitude de variações esperadas.

3 Requisitos de Software

Declara os diversos tipos de requisitos do sistema, durante o levantamento de requisitos a equipe de desenvolvimento tenta entender os domínios que devem ser automatizados pelo sistema de software, e as necessidades do mesmo (BEZERRA, 2007). Onde os

requisitos são propriedades de extrema importância e desejáveis para um sistema de software.

3.1 Requisitos Funcionais

Requisitos funcionais descrevem as funcionalidades ou serviços que se espera do sistema (funções principais do sistema) (FALBO, *et al*, 2007). Documenta como o sistema deve se manifestar a entradas específicas, como deve se comportar em determinadas situações e o que o sistema não deve fazer. Abaixo é possível visualizar os requisitos funcionais do sistema.

- RF1: O sistema deve permitir que o coordenador faça o cadastro, exclusão e alteração dos dados dos professores, com seus dados pessoais e suas informações de disponibilidade para instituição, e sua carga horária semanal;
- RF2: O sistema deverá gerar a grade de horários até que um percentual de acerto indicado pelo administrador seja válido;
- RF3: O sistema deve armazenar todas as grades de horários geradas, as utilizadas e as não utilizadas, para um histórico;
- RF4: O sistema deverá permitir a validação novamente da grade após as alterações que o coordenador fizer;
- RF5: O sistema deve evitar que o professor fique ocioso por mais de um período vago.

3.2 Requisitos Não Funcionais

Requisitos não funcionais são requisitos não diretamente relacionados às funções precípuas do sistema (FALBO, *et al*, 2007).

- RFN1: O sistema deve solicitar um usuário e senha criptografada para acesso;
- RFN2: O sistema não exige um limite de tempo para sua execução;
- RFN3: O sistema deverá dar preferência aos professores com disponibilidade de horários na parte da tarde e noite, seu nível para assumir determinadas turmas também deve ser analisado.

4 Trabalhos Correlatos

Para o desenvolvimento desse *software* foram utilizados como fonte de pesquisa trabalhos de autores baseados na tentativa de solucionar e/ou auxiliar o problema da criação da grade de horários.

4.1 Técnicas Metaheurísticas aplicadas à Construção de Grades Horárias Escolares

O trabalho realizado por Arnaldo Moura, Rafael Scaraficci, Rafael Silveira, Volnei dos Santos, em 2004, utiliza-se de três técnicas heurísticas distintas, para solucionar o problema de *timetabling*, uma evolutiva com Algoritmos Genéticos e outras duas de busca local usando busca tabu e GRASP. No decorrer do trabalho, foram gerados resultados e analisadas as técnicas aplicadas, onde alguns pontos de comparação foram definidos para essa avaliação. Assim convencionaram-se cinco classes diferentes que se baseiam na quantidade de violações das restrições dos diferentes tipos.

(A) Solução contendo apenas violações de restrições fracas, sem janelas nos horário dos professores;

(B) Solução violando apenas restrições fracas com, no máximo, três janelas e três aulas coordenadas não satisfeitas;

(C) Solução que viola, no máximo, quatro janelas, quatro aulas coordenadas e duas aulas n-uplas;

(D) Solução que viola, no máximo, cinco janelas, cinco aulas coordenadas e três aulas n-uplas;
(E) Solução infactível, ou seja, aquela que viola alguma restrição fisicamente forte (violações de indisponibilidade de horário, aulas simultâneas ou de carga-horária). Juntamente com as classes apresentadas acima foram apresentadas as restrições fortes que não podem ser violadas nunca, e as fracas que podem ser violado.

Quadro 1 – Restrições Fortes e Fracas

Fortes	Fracas
Aulas Simultâneas: que indica que um dado professor pode ministrar no máximo uma aula num dado horário disponível. Indisponibilidades: a qual significa que um professor só pode ministrar aulas caso esteja disponível no horário especificado. Aulas N-Uplas: são proibidas aulas n-uplas (n aulas no mesmo dia, seguidas ou não) para n > 2, se o número de aulas da matéria na semana for maior que 2. Caso o número de aulas seja igual a 2 são proibidas aulas duplas num dado dia.	Janelas: minimizar o número de janelas no horário dos professores. Preferências: satisfazer horários à preferência dos professores, tanto horários fixos como preferências por dias livres na semana. Aulas Geminadas: agrupar as aulas duplas existentes. Espalhamento: distribuir as aulas de maneira equilibrada durante a semana. Aulas Coordenadas: é de interesse da escola que algumas aulas sejam coordenadas entre duas ou mais turmas distintas, mantendo-as num mesmo horário.

Foi possível observar, qualidade nos resultados alcançados com algoritmo genético, mas igualmente grande dificuldade de convergência, principalmente, em relação a aulas de turmas distintas que precisam ser mantidas no mesmo horário. Já as outras técnicas específicas, embora tenham ocasionado grande melhora no algoritmo, não foram suficientes para tratar todas as restrições modeladas.

4.2 Proposta de implementação de uma ferramenta para solucionar problemas de *timetable* da UNIFRA

No referido trabalho o autor Gabriel de Senne Amorim, discente do curso de Ciência da Computação do Centro Universitário Franciscano (UNIFRA), no ano de 2012, tem por objetivo a elaboração de uma ferramenta capaz de gerar grades horárias.

Mesmo com a existência de inúmeras implementações que buscam resolver o problema de *timetable*, poucos são os que conseguem obter um resultado final. Com base neste trabalho, a análise que demonstrou os melhores resultados foi com o uso de Algoritmos Genéticos.

Apesar desses dados, foi possível perceber também que com o aumento do número de geminações no horário e com as restrições de professores e alunos se tornando mais rígidas a implementação dos algoritmos decaiu exponencialmente em desempenho.

Devido aos fatos citados, tomou-se a decisão da utilização de Algoritmos Genéticos com “dopagem” na estrutura e métodos de implementação desta ferramenta visando alcançar um melhor resultado final, considerando que a utilização da “dopagem” obteve sucesso quando utilizada por outros autores.

5 Metodologia aplicada

A utilização de uma metodologia *Agile Software Development* tem por objetivo desenvolver um software de forma rápida e descomplicada, fazendo entregas e mudanças de forma mais ágil e frequente [Cohen, Lindvall e Costa, 2003].

Existem diversos métodos de desenvolvimento todos com o objetivo de qualificar a entrega de códigos que agregue valor ao cliente por meio do desenvolvimento em pequenos ciclos [Dingsøyr, Nerur, *et al*. 2012].

Devido a seu método de desenvolvimento a metodologia escolhida para este trabalho foi a *Feature Driven Development* (FDD) [Figura 1].

O FDD baseia-se em modelos para completar uma série de funcionalidades que tem grande importância para o resultado final. Estas funcionalidades são o foco principal deste processo que através de modelos iniciais, são implementadas durante várias iterações de curta duração [Hartmann, 2005].

O FDD é dividido em cinco processos principais, onde existe uma lista de tarefas a serem executadas dentro de cada um desses processos, existindo requisitos para esta execução, algumas obrigatórias e outras opcionais [Palmer e Felsin, 2002].

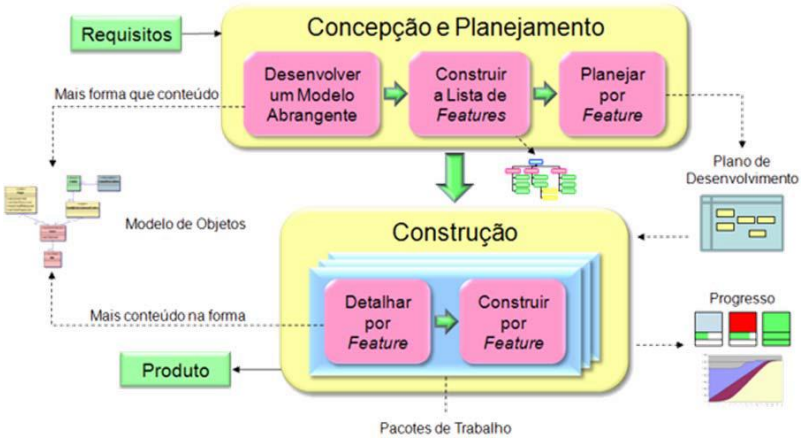


Figura 1. Processos de um projeto com FDD

6. Especificação, Projeto e Implementação

Para especificação e projeto do sistema foi utilizada a linguagem UML através do diagrama de atividade (Figura 2).

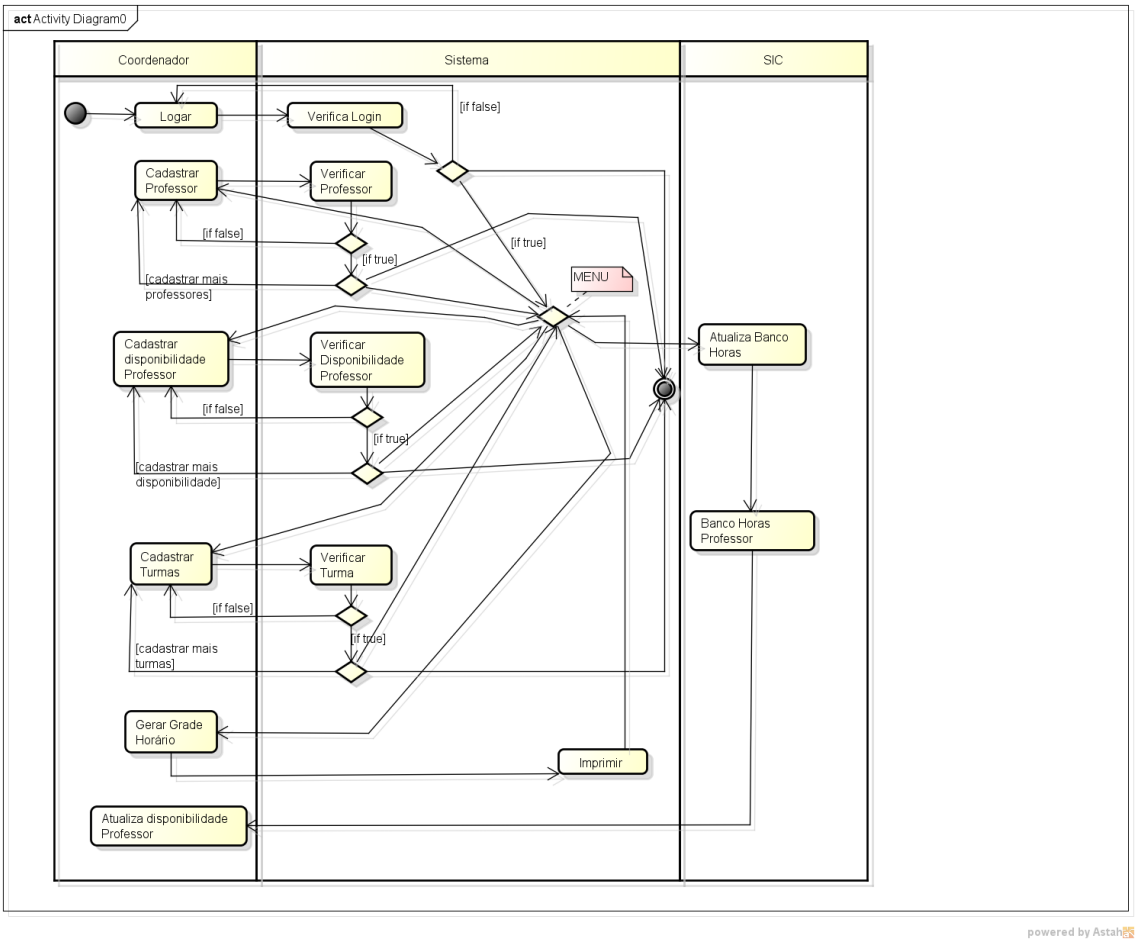


Figura 2. Diagrama de Classe

Para a implementação do *software* proposto foi utilizada a linguagem de programação PHP orientada a objeto, juntamente com recursos de desenvolvimento *web* HTML5 (HTML, CSS e JQuery) e desenvolvido utilizando uma IDE (Netbeans).

Os dados para a construção do *software* foram armazenados em um SGBD (Sistema de gerenciamento de banco de dados) *MySQL*. Os dados base para a geração da grade de horários foram inseridos manualmente via SQL. Pretende-se aprimorar o desenvolvimento do *software* permitindo que o próprio coordenador realize as inserções na grade através de *layouts*.

7 Desenvolvimento

O *software* em questão foi desenvolvido com intuito de viabilizar a construção qualificada de uma grade de horários a fim de auxiliar o coordenador pedagógico da Escola de Idiomas *Challenger Brasil*, localizada na cidade de Santa Maria (RS), na distribuição dos professores respeitando especificações pré-determinadas.

Atualmente, o quadro funcional da escola é composto por treze (13) professores todos aptos para lecionar inglês em diferentes turmas classificadas em níveis (8) e distribuídas em três categorias (*Competence; Development; Proficiency*). As categorias estão subdivididas em níveis progressivos, que foram respeitados na construção da *timetabling*, como segue: *Competence* (níveis C1, C2, C3 e C4); *Development* (D1, D2) e *Proficiency* (P1, P2). Ocasionalmente, conforme demanda, podem ser criadas novas turmas, que podem ser acrescidas na grade de horário.

As turmas são ainda distribuídas em *Slow e Regular*, sendo que o primeiro tem aula três vezes na semana em dias variados (ex.: segunda/quarta/sexta-feira; terça/quinta e sexta-feira; terça/quarta/quinta-feira) e o último de segunda à sexta-feira.

Além das especificações acima descritas, salienta-se que os professores do quadro devem ser distribuídos nas turmas conforme seu nível de proficiência na língua inglesa, avaliado pelo coordenador pedagógico. Logo, nem todos os professores estão aptos a ministrar aulas em todas as turmas. A carga horária dos docentes também foi considerada na construção do *software*, pois o sistema deve optar sempre pelo professor de maior carga horária na escola respeitando o limite de horas semanais.

Conforme metodologia da escola, não há um professor fixo para as turmas de *Competence* na semana, somente do nível D1 e subsequentes que se mantém um professor fixo.

Buscando o enquadramento da totalidade de restrições, a grade de horários da referida escola foi construída semanalmente.

7.1 Código base de implementação do *timetable*

A primeira análise realizada nesta construção foi a identificação do tipo de turma *Regular* ou *Slow*, bem como o horário de aula e o nome da turma. O nível também foi identificado e armazenado para auxiliar na escolha subsequente dos professores.

Na Figura 3, encontra-se exemplificado o demonstrativo do desenvolvimento do código base para a descrição feita.

```
if ($tipo->getNome() == "Regular") {
    $nivelTurma = $nivel->select($numeroTurmas[$aux]->nivel);
    $horarioTurma = $horario->select($numeroTurmas[$aux]->horario);
    echo '
        <tr>
            <td id="turma' . $aux . '">' . $horarioTurma->getHora() . '</td>
            <input type="hidden" id="hhorarioTurma' . $aux . '" value="' . $horarioTurma->id_horario . '"
            <td id="turma' . $aux . '">' . $numeroTurmas[$aux]->getNome() . '</td>
            <input type="hidden" id="hnivelTurma' . $aux . '" value="' . $nivelTurma->getIdNivel() . '" /
            <td id="nivelTurma' . $aux . '">' . $nivelTurma->getNomeNivel() . '</td>
    ';
    include 'regular.php';
}
```

Figura 3. Verificação de turmas *Regular/Slow*, horário, nome e nível da turma

Na sequência foi realizada a identificação do dia da semana (ex.: segunda-feira), bem como quais os professores disponíveis nesse dia. Concluída tal identificação foram selecionados os professores com disponibilidade de horário de acordo com a turma em análise. Então selecionou-se os professores capacitados de acordo com o nível da turma eliminando da listagem aqueles de nível inferior ao limite exigido como segue na Figura 4.

```
if ($contaSemana == 1) {
    $totalProf = count($numeroProfessor);
    $aux2 = 0;
    $valor = 0;
    echo '
            <td id="segunda' . $aux . '">
                <select id="seg' . $aux . '" name="seg' . $aux . '">
                    <option value="0"></option>
        ';
    while ($totalProf > 0) {
        $aux3 = 0;
        $phh = $profHor->buscaProhasHor($numeroProfessor[$aux2]->id_professor, "professor");
        $horaDisponivel = count($phh);
        while ($horaDisponivel > 0) {
            if ($phh[$aux3]->semana == $contaSemana) {
                if ($phh[$aux3]->horario == $numeroTurmas[$aux]->horario) {
                    if ($numeroProfessor[$aux2]->nivel >= $numeroTurmas[$aux]->nivel) {
                        $valor++;
                        echo '
                            <option value="' . $numeroProfessor[$aux2]->id_professor . '">'
                            . $numeroProfessor[$aux2]->nomeProfessor . '</option>
                        ';
                    }
                }
            }
            $horaDisponivel--;
            $aux3++;
        }
        $totalProf--;
        $aux2++;
    }
```

Figura 4. Identifica dia da semana, professores disponíveis no horário e nível do professor em relação à turma

Seguindo o exposto no trecho de código anterior onde os professores disponíveis foram selecionados, o código em sequência (Figura 5) demonstra a busca do professor de maior carga horária. Ao identificá-lo o *software* irá aloca-lo como primeira opção, no entanto criou-se a possibilidade de o coordenador alterar tal definição conforme desejar.

```
maiorCarga = 0;
prof = "";
idProf = "";
$('#qua' + conta).find('option').each(function() {
    if (parseInt(maiorCarga) < parseInt(carga[$(this).val()])) {
        if (maiorCarga != 0) {
            $('#qua' + conta + ' option:selected').prop('selected', false);
        }
        maiorCarga = carga[$(this).val()];
        idProf = $(this).val();
        prof = $(this).text();
        $("#qua" + conta + " option[value='" + idProf + "']").attr("selected", "true");
    }
});
carga[$('#qua' + conta).val()]--;
```

Figura 5. Busca por professor disponível e maior carga horária

Na Figura 6 foi realizada a busca do nível da turma que havia sido armazenado anteriormente, avalia-se o nível e é possível identificar se ele é maior ou igual (≥) que o nível D1. Caso isso aconteça irá ocorrer a busca do professor de maior carga horária entre os selecionados. Com a confirmação o nível ≥ a D1 ele automaticamente será alocado em todos os dias da semana. Vale lembrar que conforme descrito anteriormente

o níveis inferiores a D1 possuem professores diferentes a cada dia de aula, com variações.

```
function gerar(conta) {
    if ($('#hnivelTurma' + conta).val() >= 5) {
        maiorCarga = 0;
        prof = "";
        idProf = "";
        $('#seg' + conta).find('option').each(function() {
            if (parseInt(maiorCarga) < parseInt(carga[$(this).val()])) {
                if (maiorCarga != 0) {
                    $('#seg' + conta + ' option:selected').prop('selected', false);
                }
                maiorCarga = carga[$(this).val()];
                idProf = $(this).val();
                prof = $(this).text();
                $("#seg" + conta + " option[value='" + idProf + "']").attr("selected", "true");
            }
        });
        if ($("#ter" + conta + " option[value='" + idProf + "']").text() == prof) {
            $("#ter" + conta + " option[value='" + idProf + "']").attr("selected", "true");
            carga[$('#seg' + conta).val()]--;
        }
        if ($("#qua" + conta + " option[value='" + idProf + "']").text() == prof) {
            $("#qua" + conta + " option[value='" + idProf + "']").attr("selected", "true");
            carga[$('#seg' + conta).val()]--;
        }
        if ($("#qui" + conta + " option[value='" + idProf + "']").text() == prof) {
            $("#qui" + conta + " option[value='" + idProf + "']").attr("selected", "true");
            carga[$('#seg' + conta).val()]--;
        }
        if ($("#sex" + conta + " option[value='" + idProf + "']").text() == prof) {
            $("#sex" + conta + " option[value='" + idProf + "']").attr("selected", "true");
            carga[$('#seg' + conta).val()]--;
        }
```

Figura 6. Avalia nível da turma e insere professor de maior carga horária

Último trecho de código (Figura 7), após todos os passos anteriormente descritos, demonstra que o *software* não permitirá a presença do mesmo professor no mesmo horário em duas turmas diferentes no mesmo dia, logo ele removerá o professor alocado de todas as outras turmas em mesmo horário e dia da semana. Mas poderá realocá-lo no mesmo dia da semana desde que em um horário diferente.

```
while(aux<totalturmas){
    if(horario==$("#hhorarioTurma"+aux).val() && conta!=aux){
        if($("#seg" + aux + " option[value='" + idProf + "']")){
            $("#seg" + aux + " option[value='" + idProf + "']").remove();
        }
        if($("#ter" + aux + " option[value='" + idProf + "']")){
            $("#ter" + aux + " option[value='" + idProf + "']").remove();
        }
        if($("#qua" + aux + " option[value='" + idProf + "']")){
            $("#qua" + aux + " option[value='" + idProf + "']").remove();
        }
        if($("#qui" + aux + " option[value='" + idProf + "']")){
            $("#qui" + aux + " option[value='" + idProf + "']").remove();
        }
        if($("#sex" + aux + " option[value='" + idProf + "']")){
            $("#sex" + aux + " option[value='" + idProf + "']").remove();
        }
    }
```

Figura 7. Remove professor de outras turmas quando escolhido para o mesmo horário

8 Resultados

Após a exposição dos códigos para o desenvolvimento do *software* deste trabalho, a seguir são mostradas capturas de imagens da grade de horário gerada. Na Figura 8 é mostrada a visão da página inicial da aplicação onde são fornecidos o horário em que haverá turmas, o seu nome (ex.: nome da turma 8C) e o respectivo nível, após buscar todas as turmas em andamento dentro da escola.

O *software* distribuirá os dias em que as turmas terão aulas, pois como explicado anteriormente estas podem ser, quanto frequência de dias letivos, *Regular* ou *Slow*, sendo que o último o não terá aula todos os dias como no primeiro.

Após coletado essas informações o software ainda faz a busca dos horários em que os professores se encontram disponíveis. Com a realização dessa busca o software ainda não aloca professores na grade para que não fique confuso para o usuário. Porém, com o primeiro dado sempre vazio, o sistema posiciona os professores nos horários em que existem turmas.

Para melhorar a criação da grade o *software* compara todos os professores que tem disponibilidade no horário com o nível da turma em questão, avaliando se o professor está capacitado para devida turma, assim deixando disponível apenas os professores que realmente poderiam dar aula para esta.

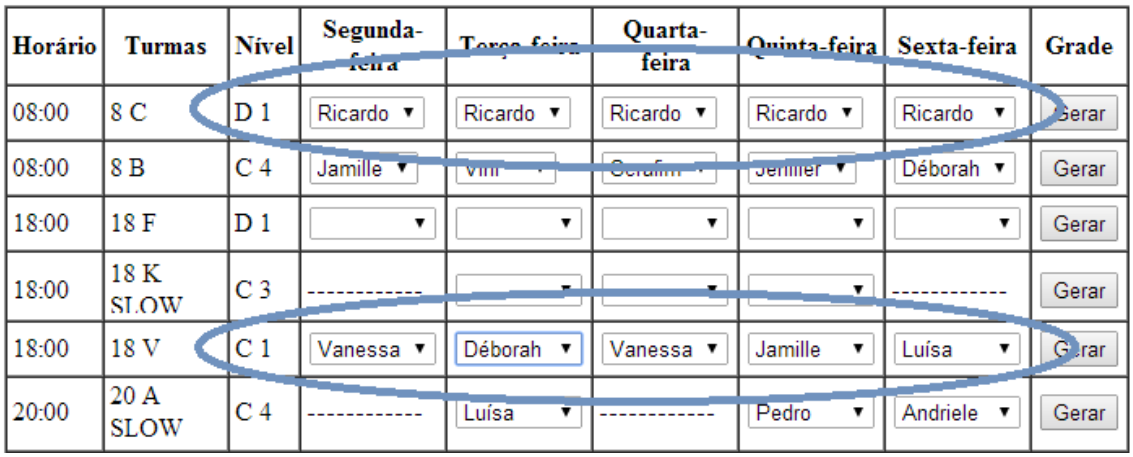
Ao final de cada linha, encontra-se disponível um botão para que o coordenador possa gerar automaticamente a inserção do professor para a turma, essa execução está baseada em mais uma das regras da escola, dentre os disponíveis o professor de maior carga deve ter a preferência. Ao inserir o professor, este é excluído das demais turmas que estejam no mesmo horário e dia da semana em que ele já foi selecionado. Assim o resultado da grade é gerado automaticamente, mas mesmo assim há possibilidade do coordenador realizar as alterações manualmente.

Horário	Turmas	Nível	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira	Grade
08:00	8 C	D 1	Ricardo	Ricardo	Ricardo	Ricardo	Ricardo	Gerar
08:00	8 B	C 4	Jamille	Vini	Serafim	Jeniffer	Déborah	Gerar
18:00	18 F	D 1						Gerar
18:00	18 K SLOW	C 3	----------				----------	Gerar
19:00	19 D	P 1	Vanessa	Vanessa	Vanessa	Vanessa	Vanessa	Gerar
20:00	20 A SLOW	C 4	----------	Luísa	----------	Pedro	Andriele	Gerar

Figura 8. *Software* de geração da grade de horário

Ao utilizar o botão para geração automática, além da busca do professor de maior carga horária o *software* já identifica o nível da turma, pois se a turma for de nível igual ou superior ao D1 o mesmo professor deverá ministrar as aulas a semana inteira (linha 1 da grade – Figura 9).

Na seleção realizada na linha 5 da Figura 9 é demonstrado como o *software* procede quando o nível da turma é inferior a D1, nesse caso professores diferentes são alocados nos dias da semana para ministrar as aulas. Salientando que podem haver repetições como no caso abaixo.



Horário	Turmas	Nível	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira	Grade
08:00	8 C	D 1	Ricardo	Ricardo	Ricardo	Ricardo	Ricardo	Gerar
08:00	8 B	C 4	Jamille	Vini	Serafim	Jenifer	Déborah	Gerar
18:00	18 F	D 1						Gerar
18:00	18 K SLOW	C 3	----------				----------	Gerar
18:00	18 V	C 1	Vanessa	Déborah	Vanessa	Jamille	Luísa	Gerar
20:00	20 A SLOW	C 4	----------	Luísa	----------	Pedro	Andriele	Gerar

Figura 9. Demonstração de diferenciação de níveis/turmas

Na sequência, a Figura 10 demonstra a distribuição de um mesmo professor (grifo do autor – linhas 1 e 3/coluna 8) no mesmo dia da semana, em turmas e níveis diferentes, sendo utilizado em horários alternados conforme disponibilidade previamente definida.

A distribuição dos professores no *software*, é feita através da Força Bruta. Onde o *software* durante a criação da grade vai realizando uma busca professor por professor e inserindo (disponibilizando) para a inserção na grade apenas os professores que estiverem capacitados a ministrar a aula em determinada turma.

É mostrada também, nas linhas 6 e 4/colunas 4 e 5 respectivamente, a seleção de professores no banco de dados do *software*, com disponibilidade de horário e capacitação para a turma, a fim de facilitar a busca pelo coordenador.

Na linha 10 (Figura 10) salienta-se um dos benefícios da utilização do *software*, pois no caso circulado mesmo o nível da turma sendo superior a D1, o que definiria um mesmo professor para todos os dias de aula da turma, há um dia em que não poderá ser utilizado o mesmo professor devido este já ter sido excluído pelo *software* no momento em que foi identificado ausência de disponibilidade (não possui horário livre em tal dia).

Figura 10. Especificações do software

9 Conclusões Finais e Trabalhos Futuros

O problema de confecção de uma grade de horário deve ser tratado segundo o tipo de escola e seu sistema educacional. Este problema tem sido extensivamente estudado ao longo das últimas décadas e sabe-se que, conforme observado durante a construção desse trabalho, muitas soluções já foram propostas com o intuito de solucioná-lo.

Considera-se que o presente trabalho atingiu as expectativas primárias almejadas no entendimento que os objetivos propostos foram alcançados, onde o *software* encontra-se em utilização no auxílio do desenvolvimento da grade semanal.

Foi proposto inicialmente o desenvolvimento de um *software* capaz de auxiliar na construção da grade de horário da escola de idioma. Através das considerações necessárias acerca das restrições impostas, obteve-se então o resultado final, uma *timetable* que se encontra em teste na escola e auxiliando o coordenador pedagógico na distribuição de seus professores.

Citando os trabalhos correlatos, o primeiro que executou sua implementação através de Algoritmos Genéticos encontrou respostas, mas não obteve sucesso tendo dificuldade em organizar turmas diferentes que precisavam ser mantidas no mesmo horário.

Já o segundo, apresentou penalidades que inviabilizam a possibilidade de se ter um resultado preciso, pois o algoritmo genético depende da entrada do problema, desta forma, podendo existir várias configurações para o mesmo problema.

Salienta-se que, embora o tempo computacional do *software* desenvolvido seja considerado moderado (aproximadamente 3 minutos na máquina da escola) baseado no

número de consultas SQL, o mesmo tem reduzido o tempo gasto pelo coordenador para construção da grade em comparação ao gasto anteriormente sem o *software*.

Algumas propostas de trabalhos futuros são apresentadas a seguir. São indicações ou de continuidade do relato aqui exposto ou novas implementações, não abordadas no desenvolvimento desse trabalho de pesquisa.

Futuramente considera-se como possibilidade para novas implementações aprimorar o *software* de maneira que ele apresente maior interação com o usuário. Através da criação de *layouts* seria possível permitir que a inserção de informações seja feita pelo próprio responsável pela geração da grade de horário.

A utilização do *software* foi planejada para ser executada de forma local e não na *web*, ou seja, para sua utilização os dados e informações deverão ser instalados em uma máquina que deverá ser preparada para tal criação. Considera-se a possibilidade de desenvolvimento para viabilizar a execução do *software* também via *web*.

Para agregar futuras melhorias pretende-se que o *software* seja desenvolvido baseado em um algoritmo genético para verificar se ocorrerão melhorias em seu tempo de execução, pois o *software* desenvolvido apresenta um número elevado de testes o que exige muitas consultas SQL no banco de dados.

Considera-se, igualmente, a possibilidade de que através do *tuning* nas instruções SQL o tempo de resposta e recuperação de dados poderia ser minimizado, melhorando o acesso e otimizando a transferência de dados.

Referências Bibliográficas

Aguiar, M.S. (1998). Análise formal da complexidade de algoritmos genéticos. Universidade Federal do Rio Grande do Sul. Instituto de Informática. Curso de Pós Graduação em Ciência da Computação.

Bezerra, Eduardo. Princípios e análise e projeto de sistemas com UML. 2ª ed. Rio de Janeiro: Elsevier, 2007.

Caldeira, J. C.; Rosa, A, C. (1997). School Timetabling using Genetic Search, PATAT (1997), pp. 115-122.

Camargo, A.M; Carobo, C.B; Baggio, M.A et al. (2011). Alocação de professores usando escalonador de timetabling swictboard. IMED. Passo Fundo (RS).

Cormen, T.H; Leiserson,C.E; Rivest, R.L; Stein, C. (2002). Algoritmos: Teorias e prática. Campus, Tradução da 2º edição americana.

Cohen, D; Lindvall, M; Costa, P. (2003). Agile Software Development. DACS State-of-the-Art/Practice Report, Fraunhofer Center Maryland.

Dingsøyr, T. et al. (2012). A decade of agile methodologies: Towards explaining agile software development. The Journal of Systems and Software.

Falbo, R.A., Martins, A.F.; "Um processo de Engenharia de Requisitos Baseado em Reutilização e padrões de Análise. VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC'07), Lima, Peru, 2007.

Fucilini, T.P; Maruari, E; Rebonatto, M.T. (2008). Timetabling com Algoritmos Genéticos: Resultados, Restrições e Exploração do Paralelismo. Curso de Ciência da Computação. Universidade de Passo Fundo (UPF). Hífen, Uruguaiana, v.32 – nº62 – II semestre – Ano 2008 – ISSN 1983-6511.

Hamawaki, C.D.L. (2005). Geração Automática de Grade Horária Usando Algoritmos Genéticos: O Caso da Faculdade de Engenharia Elétrica da UFU. Uberlândia. 104 p. Originalmente apresentada como dissertação de Mestrado, Universidade Federal de Uberlândia.

Hartmann, J. (2005). Utilizando Padrões Organizacionais e Avaliação de Risco para Adaptar a Metodologia de Desenvolvimento de Software. LUME – Repositório Digital Universidade Federal do Rio Grande do Sul. Disponível em: <http://hdl.handle.net/10183/5651>. Acesso em: 20 abr.

Holl, Rodrigo. Metodologia ágil de desenvolvimentos de software de FDD. Acesso em: <http://htwojsystem.wordpress.com/2011/06/26/metodologia-agil-dedesenvolvimento-de-software-fdd/.

Koerich, A.L. (2004). Força Bruta: Técnica de Projeto de Algoritmos. Ciência/Engenharia de Computação. Proj. Anl. Algoritmos. PUCPR.

Linden, R. (2008) Algoritmos genéticos. 2ª ed. Rio de Janeiro: Brasport.

Lourenço, H. R.; Paixão, J. P.; Portugal, R. (2001) Multiobjective metaheuristics for the bus-driver scheduling problem. Tranportation Science, v. 35, p. 331-343.

Moura, A; Scaraficci, R; Silveira, R; Santos, V. Técnicas Metaheurísticas aplicadas à Construção de Grades Horárias Escolares.

Palmer, S. R.; Felsin, J. M. (2002). A Practical Guide to Feature-Driven Development. Prentice Hall.

Quadros, M.A.C. (2011). Métodos de Captura de informações de usuário. Universidade Tecnológica Federal do Paraná –UTFP. Curitiba.

Soares, L.G. (1997). Algoritmos genéticos: estudo, novas técnicas e aplicações. Minas Gerais (MG).

Terra, I.P; Radaelli, J.L. Utilização dos métodos de otimização em problemas de timetabling. Principium online: Iniciação Científica no Unileste-MG, Coronel Fabriciano, v. 1, n. 1, p. 95-103, jul. 2007.

Timóteo, G.T.S. (2002). Desenvolvimento de um Algoritmo Genético para Resolução do Timetabling. Universidade Federal de Lavras – MG. Monografia.